Porting code

2 different challenges:

- Stencil from CGG
  - Simple code 143 lines (but representative)
  - Make it run faster
- Code Saturn from EDF
  - Much more realistic code
  - Port the code to ARM

CGG Stencil code

COMPILE:

```
g++ stencil.cxx -O3 -o stencil
```

RUN:

```
./stencil 100 100 100 5
```

CHALLENGE:

The challenge is to modify the code, with the goal of running as fast as possible, whilst still producing the same results.

CGG Stencil code

OUTPUT:

The first columns are for validation to verify that the different tunings produce the same output

Performance will be judged on column 7 : the time in microseconds for 1 iteration

```
[ec2-user@ip-172-31-12-167 stencil]$ ./stencil 100 100 100 5
_0_   0.945022 -1.268302  0.321698  0.789154 -1.060457    2182729    2182.729 100 100 100
_0_   1.191360  1.527541  0.268107  0.768831  1.102787    2180140    2180.140 100 100 100
_0_   1.184568 -1.795619  0.037173  0.647730 -1.141783    2179131    2179.131 100 100 100
_0_   1.437763  2.067386  0.132803  0.610521  1.179974    2174182    2174.182 100 100 100
.....

Output simplified for clarity.
```

CGG Stencil code – Getting Started

stencil.tgz

Filename	What is it?
stencil.cxx	The source code
stencil	An executable (just for reference)
stencil.sh	A script to run the executable
compile.sh	A script to compile the executable
stencil-big-test.sbatch	A script to submit a small test to Slurm
stencil-small-test.sbatch	A script to submit a big test to Slurm

CGG Stencil code

- **NOTES:**
- Parameters are “X dim”, “Y dim”, “Z dim” and number of iterations
- 3 iterations are sufficient to observe a stable performance
- The real code uses multiple kernels and multiple matrices; they all expose the same challenges and we use globally the same tricks to achieve peak performance on any multicore processor
- We suggest starting with small dimensions, then scaling up as the code performance improves. To make the challenge more realistic, performance at larger sizes is of interest.
- Expect code improvements to be more visible at larger sizes.
- The performance will be judged with the following input parameters 1000 1000 1000 5
- Note that the memory consumption will increase as you increase the scale. Ensure when you test at 1000x1000x1000 you select the big queue.
- The team with the fastest time will get 50 points, next fastest 48, then 46…….